Stephanie Fissel

Jun 3, 2021 • 5 min read

Databases: Discussing Popular Architectures

By Hayden French, Katlyn Walter, Silas Hayes, Stephanie Fissel

After a week of designing a set of our own data, populating them into three different database architectures (Neo4j, MongoDB, and MySQL), and writing python scripts to query each of them, we have developed a better understanding of what databases really are and how to manipulate our data to fit each respective architecture. Databases are crucial to any company or organization, which is why we wanted to share the insights we have gained.

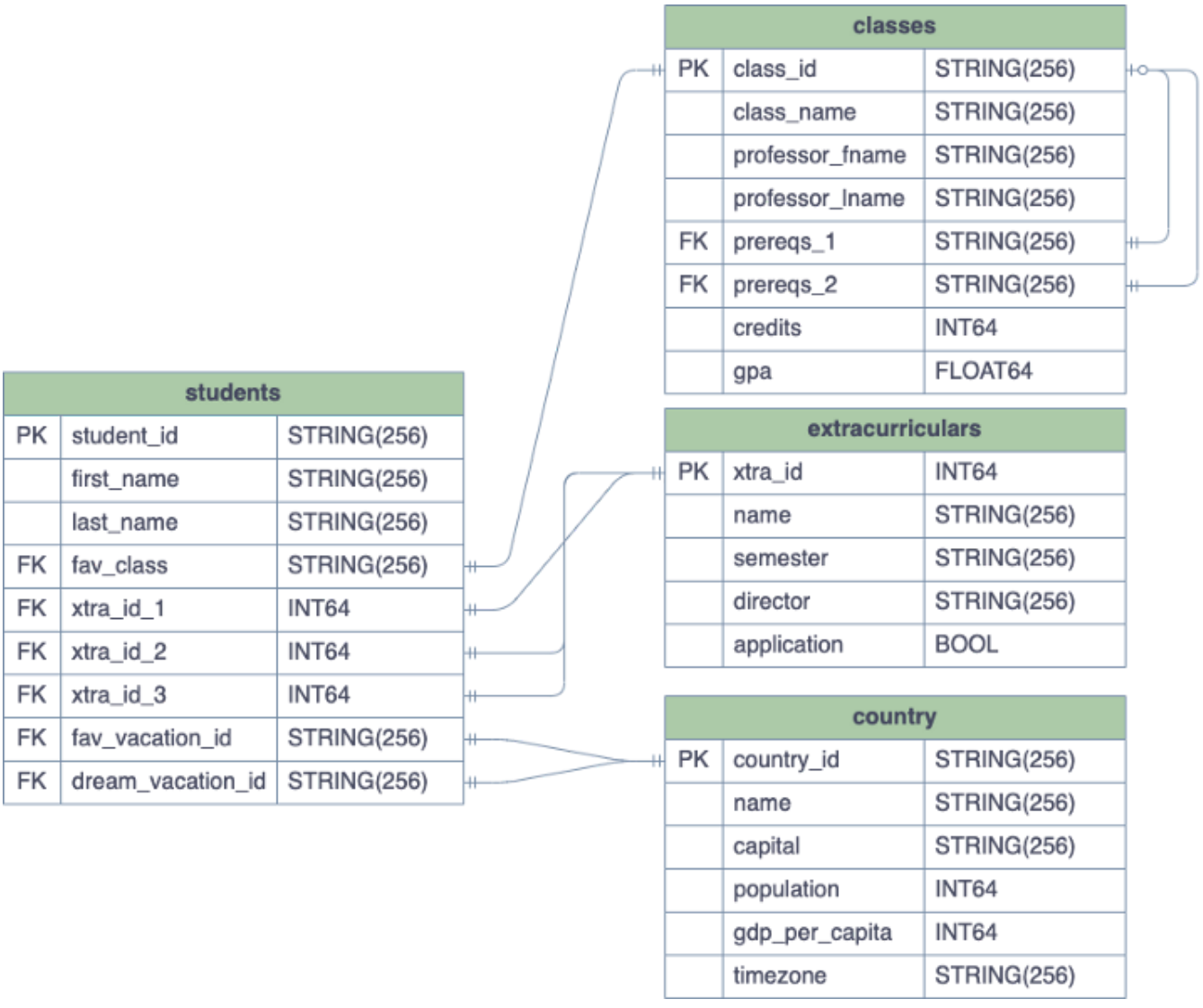
. . .

To start, a database is a collection of organized data. Assuming the database is well-maintained, it can be extremely useful for storing a large amount of data while keeping it easy to manipulate. Another advantage of databases is that they can exist on remote servers meaning that they can be very large while still accessible by multiple people. Finally, databases can require credentials, meaning sensitive data (e.g., your passwords and other personal information) is protected.

Database Categories

There are multiple options when it comes to database models, all of which have different logical structures for data storage. Some of the most popular database models include the relational model, the document model, and the graph model.

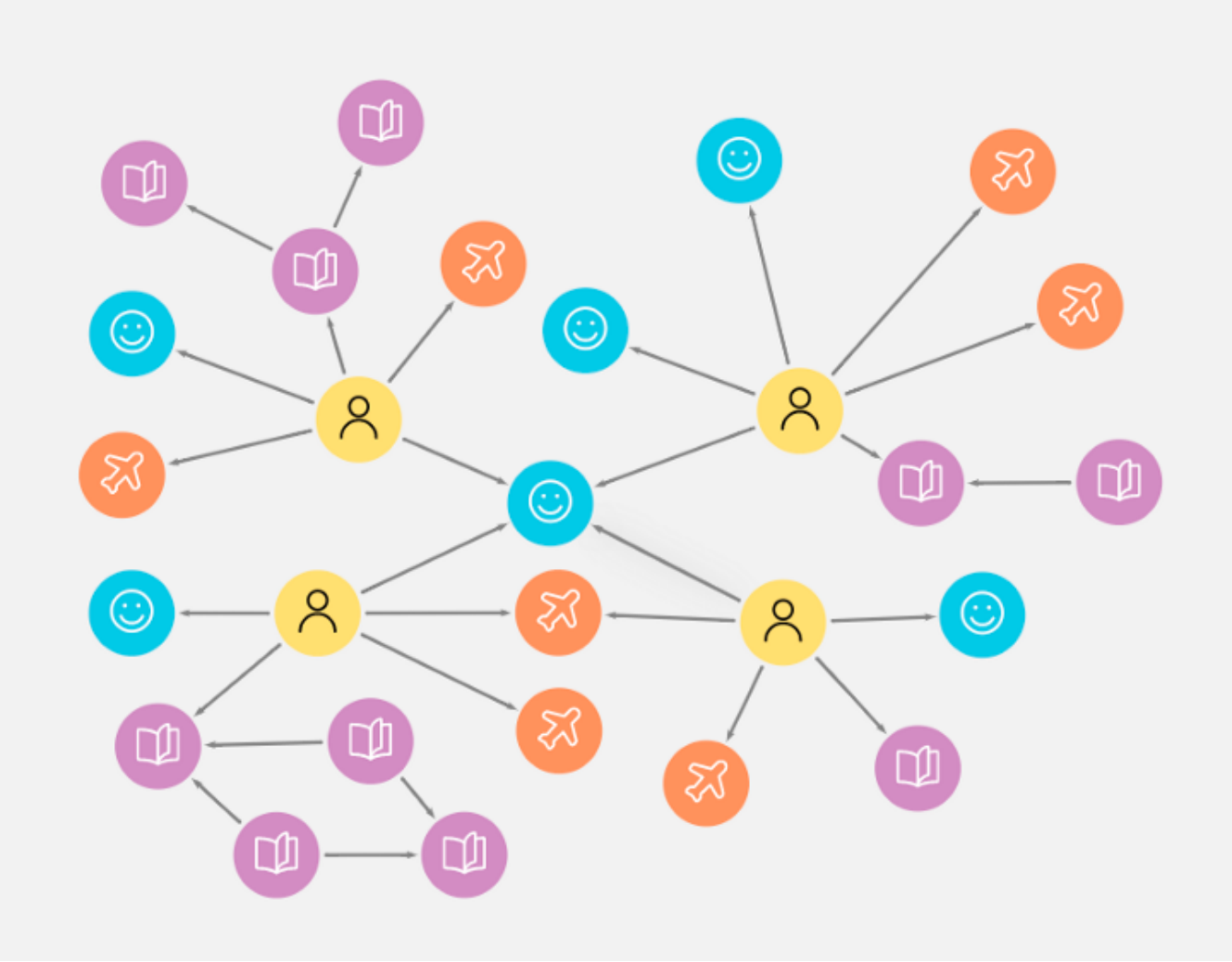
A relational model, such as MySQL, stores data in multiple tables with observations connected to each other by keys. Each table contains related data on the subject. For example, in the diagram below the extracurriculars table includes the name, which semester the extracurricular is active for, the director of it, and whether it requires an application. The extracurriculars table then connects through a key to the students table, which contains information concerning which extracurriculars each student participates in. These are the data we will be working with for the other two databases as well; however, each will interact with the data in different ways.



Document models, like MongoDB, store each data item separately from each other. These data items (documents) are not related to each other as they are in the relational database and can have multiple different properties within them.

A good analogy for a document database is a library. Documents (books) are given unique ids (Dewey Decimal call numbers) and each document can contain a variety of information (pages). For our data, this looks like having one document for each student. Inside that document is nested information that contains data on all of the classes, extracurriculars, and travel countries relating to that student. For instance, within the extracurriculars section will be the info on semesters, director, and application.

A **graph model**, like Neo4j, takes the rows from our tables and makes each one into a node, colored by which table it belongs to and stores the information of the row. Those nodes can then be connected to others through named relationships, which can sometimes have properties depending on the graph model. This model can be exceptionally useful when modeling something like a social network, as it is easy to see who interacts with whom and what shared interests connect users.



Editing for Each Database

For MySQL, we didn't have to edit our data at all. This is because we entered our data with a relational database in mind, and MySQL is highly integrated with pandas.

For Neo4j, we were able to add some more details to our data. This is because, as a graph database, Neo4j allows us to structure the data as nodes and the relationships between them. More specifically, we were able to include additional information

about the relationships between nodes. For example, we have a Hayden node and a marching band node. With Neo4j, we can add extra data to that relationship, like Hayden has participated in the marching band for 2 years. With our relational database, it wouldn't make sense to include that information under the extracurricular table since it is not an inherent property of the marching band activity. However, it is a property of Hayden's relationship to the marching band so we can include that information there.

Finally, for MongoDB we tweaked the structure of the data but not so much the data itself. This is because Mongo is a document model of database and our data was in a relational form. All we had to do was to rearrange it, creating a nested form of data within data where we would have referenced another table in MySQL or node in Neo4j.

Pros & Cons of Each Database

- Pros: Highly rigid structure, easy to connect related tables
- Cons: Highly rigid structure, can be hard to include new data

- Pros: Document-based storage allows for more flexibility, easy to scale for large datasets
- Cons: Can't connect related collections, can be difficult to transfer data to (depends on the type)

- Pros: Unique visualizations, inclusion of relationships allowed us to include additional data
- Cons: Graph visualizations can get complicated very quickly, difficult to use for large datasets, can be difficult to query

• • •

Key Terms:

CSV: A file extension that stands for comma separated values. One of the most popular formats for tabular data.

Database: An organized collection of data.

Relational database: A database structured around the relations between stored items. A popular example is MySQL.

Document database: A database structured around storing data in JSON-like documents. A popular example is MongoDB.

Graph database: A database structured around nodes (an entity) and a relationship. A popular example is Neo4j.

SQL: Stands for Structured Query Language, essentially a standardized language used for passing queries to some types of databases. (Not to be confused with MySQL, which is a specific database which utilizes SQL)

Query: A request passed to a database. Usually done to retrieve data, but can be done to edit or even delete data as well.

Key: An attribute that helps identify a row in a table. Can be primary (a unique identifier for each row) or foreign (when referred to in the context of another table).

Pandas: A popular data analysis package for python.